Structures

C programming

Structures

A structure is a user defined data type available in C that allows to combine data items of different kinds.

Structure declaration

- You can define a structure in the global scope of your program (outside of all your functions, just like the functions prototypes).

- You can declare elements of your structure in its scope.

```c
struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  return (0);
}
```

Structures

You can access the elements of your structure by using the ‘.’ symbol.

```c
struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  user.name = "Foo Bar";
  return (0);
}
```

Example - step 0

Address	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Variable																				
Value	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Address	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Variable																				
Value	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

```c
#include <stdio.h>

struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  user.name = "Foo Bar";
  user.email = "foo@hbtn.io";
  user.age = 98;
  return (0);
}
```

Example - step 1

Address	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Variable	"Foo Bar"								"foo@hbtn.io"											
Value	F	o	o	SPC	B	a	r	\0	f	o	o	@	h	b	t	n	.	i	o	\0
Address	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Variable																				
Value	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

```c
#include <stdio.h>

struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  user.name = "Foo Bar";
  user.email = "foo@hbtn.io";
  user.age = 98;
  return (0);
}
```

Example - step 2

Address	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Variable	"Foo Bar"								"foo@hbtn.io"											
Value	F	o	o	SPC	B	a	r	\0	f	o	o	@	h	b	t	n	.	i	o	\0
Address	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Variable	user																			
	user.name								user.email								user.age			
Value	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

```c
#include <stdio.h>

struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  user.name = "Foo Bar";
  user.email = "foo@hbtn.io";
  user.age = 98;
  return (0);
}
```

Example - step 3

Address	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Variable	"Foo Bar"								"foo@hbtn.io"											
Value	F	o	o	SPC	B	a	r	\0	f	o	o	@	h	b	t	n	.	i	o	\0
Address	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Variable	user																			
	user.name								user.email								user.age			
Value	20								28								98			

```c
#include <stdio.h>

struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  user.name = "Foo Bar";
  user.email = "foo@hbtn.io";
  user.age = 98;
  return (0);
}
```

Pointers to structures

To access elements of a pointer to a structure, you have to dereference the pointer and then access to the data using the ‘.’ symbol.

BUT

There is a simple way to do that, just by using the ‘->’ symbol.
This symbol is equivalent to dereferencing + using ‘.’

```c
struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;
  struct User *ptr;

  ptr = &user;
  /* Dereferencing the pointer before
     access the data with the '.' symbol */
  (*ptr).name = "Foo Bar";
  /* Using the '->' works the same, and
     is so much easier */
  ptr->email = "foo@hbtn.com";
  return (0);
}
```

Exercise

Complete the function “new_user” to get the following output:

Output:

```
[Alex@Alexandres-MacBook-Pro:~/Holberton/Projects$ ./a.out
User Foo created !
His email is: foo@foo.bar
And he is 98 years old
Alex@Alexandres-MacBook-Pro:~/Holberton/Projects$
```

```c
#include <stdlib.h>
#include <stdio.h>

struct User
{
  char *name;
  char *email;
  int age;
};

struct User *new_user(char *name, char *email, int age)
{
  struct User *user;

  /* Complete this function */

  return user;
}

int main(void)
{
  struct User *user;

  user = new_user("Foo", "foo@foo.bar", 98);
  if (user == NULL)
    return (1);
  printf("User %s created !\n", user->name);
  printf("His email is: %s\n", user->email);
  printf("And he is %d years old\n", user->age);
  return (0);
}
```

Exercise - answer

Output:

```
[Alex@Alexandres-MacBook-Pro:~/Holberton/Projects$ ./a.out
User Foo created !
His email is: foo@foo.bar
And he is 98 years old
Alex@Alexandres-MacBook-Pro:~/Holberton/Projects$
```

```c
#include <stdlib.h>
#include <stdio.h>

struct User
{
  char *name;
  char *email;
  int age;
};

struct User *new_user(char *name, char *email, int age)
{
  struct User *user;

  user = malloc(sizeof(struct User));
  if (user == NULL)
    return (NULL);
  user->name = name;
  user->email = email;
  user->age = age;
  return user;
}

int main(void)
{
  struct User *user;

  user = new_user("Foo", "foo@foo.bar", 98);
  if (user == NULL)
    return (1);
  printf("User %s created !\n", user->name);
  printf("His email is: %s\n", user->email);
  printf("And he is %d years old\n", user->age);
  return (0);
}
```